OR Spectrum (2017) 39:749–773
DOI 10.1007/s00291-017-0474-2

REGULAR ARTICLE

Resource allocation problems in decentralized energy management

Thijs van der Klauw[1] · **Marco E. T. Gerards**[1] · **Johann L. Hurink**[1]

Received: 7 July 2016 / Accepted: 27 January 2017 / Published online: 13 February 2017

Abstract Changes in our electricity supply chain are causing a paradigm shift from centralized control towards decentralized energy management. Within the framework of decentralized energy management, devices that offer flexibility in their load profile play an important role. These devices schedule their flexible load profile based on steering signals received from centralized controllers. The problem of finding optimal device schedules based on the received steering signals falls into the framework of resource allocation problems. We study an extension of the traditional problems studied within resource allocation and prove that a divide-and-conquer strategy gives an optimal solution for the considered extension. This leads to an efficient recursive algorithm, with quadratic complexity in the practically relevant case of quadratic objective functions. Furthermore, we study discrete variants of two problems common in decentralized energy management. We show that these problems are NP-hard and formulate natural relaxations of both considered discrete problems that we solve efficiently. Finally, we show that the solutions to the natural relaxations closely resemble solutions to the original, hard problems.

This research is conducted within the EASI project (12700) supported by STW and Alliander and the EU FP7 project e-balance (609132).

✉ Thijs van der Klauw
t.vanderklauw@utwente.nl

Marco E. T. Gerards
m.e.t.gerards@utwente.nl

Johann L. Hurink
j.l.hurink@utwente.nl

[1] Department of Electrical Engineering, Mathematics and Computer Science, University of Twente, Drienerlolaan 5, 7522NB Enschede, The Netherlands

Keywords Resource allocation · Convex optimization · Decentralized energy management

1 Introduction

Our electricity supply chain is changing rapidly. Traditionally electricity is supplied by a small number of large producers that are centrally controlled. The goal of the current control methodology is to ensure the required balance between production and consumption at all times by letting production follow the demand. However, driven by environmental targets, a large number of small-scale generation units are being introduced in the system in recent years, many of which exploit renewable, uncontrollable sources such as wind and sun (Gönsch and Hassler 2016). To offset the loss of flexibility on the production side and avoid very high investment requirements in infrastructure, flexibility on the demand side is increasingly considered as a valuable alternative (Siano 2014; Vardakas et al. 2015). This flexibility on the demand side comes from appliances that are capable of changing their load profile without significantly decreasing user comfort. Examples are an electric vehicle (EV) shifting its charging from the evening to the night or a heat pump combined with heat storage which produces heat using electricity before it is needed and storing it in the heat storage. Many of these devices are, if not already present inside the homes of residential consumers, expected to be introduced in large numbers in the coming years.

To properly use the flexibility provided by electric appliances, specifically those used by residential customers, the traditional centralized control methodologies do not suffice as they do not scale to a large number of entities. Furthermore, it is hard to properly deal with the large diversity of devices. Thus, new approaches are required to schedule the use of flexibility of devices emerging on the customer side of the grid.

In recent years, several new approaches have been suggested in the literature. Some of these approaches require the devices to communicate their available flexibility to a central controller (e.g. the PowerMatcher (Kok 2013) and the Intelligator (Claessens et al. 2012)). However, such a centralized approach requires the disclosure of privacy-sensitive information. Furthermore, to be able to solve the central problem simplifying assumptions are generally made. For example, the PowerMatcher does not look ahead to potential future requirements of flexibility and can hence use up flexibility before the most opportune time. The Intelligator, on the other hand, aggregates the constraints for the individual devices into a single flexibility constraint on the central level, which can potentially cause the resulting solution to be infeasible on the device level. Furthermore, the approach requires the use of self-learning techniques to determine the relevant parameters. In case the parameters are hard to learn, the system operates sub-optimally (Claessen et al. 2014).

Another approach is decentralized energy management (DEM). In such a decentralized approach, a centralized control steers the electricity consumption and production of devices through the use of steering signals. The flexible devices schedule their load profile, using their available flexibility, to best match the received signal. A popular steering signal is so-called time-of-use pricing, where the price of electricity varies over predetermined time intervals to incentivize customers to shift their consumption

to times with abundant production (see, e.g.Telaretti et al. 2016; You et al. 2012). However, such (linear) pricing structures often suffer from drawbacks in that they often only shift the peak instead of lowering it or even cause new, higher peaks in case a large number of (automated) devices respond to the same price signal (Barbato and Capone 2014; McKenna and Keane 2014). Therefore, we believe more sophisticated steering signals are required to design a system that adequately uses the available flexibility from devices. Examples of approaches that use more sophisticated signals are Gerards et al. (2015), Mohsenian-Rad et al. (2010) and Gan et al. (2013).

The approaches mentioned above, which use more general steering signals, typically deal with design of the approach on the level of the central controller (i.e. the decision on what the steering signal is going to be). However, the response of the devices to these steering signals is generally left as an open question. Furthermore, these responses are often assumed to be optimal with respect to the steering signal, to be able to prove certain properties of the approach (e.g. the convergence proved by Gan et al. (2013)). The device-level problems are generally formulated as nonlinear optimization problems that can be solved by general-purpose nonlinear solvers. However, we note that the local hardware on which such a problem has to be solved is often limited (Molderink et al. 2010). Furthermore, the device-level problems have similar structures in many cases, which allows for specific solution methods that significantly reduce hardware requirements and increase the solution speed. The latter greatly benefits approaches where many device-level problems have to be solved (e.g. the approach by Gerards et al. (2015)).

For practical applications, it is not desirable that one has to design a new scheduling method for each new variant of a device. To avoid this, a higher abstraction of device classes is needed. One attempt to define a framework for DEM with abstract device classes in the EF-PI platform developed by the Flexiblepower Alliance Network (2016). Therefore, we argue in this work that the device-level scheduling problems of many devices are in fact very similar and can be solved by the approaches we present.

In this work, we consider abstract models of several devices and argue that their load scheduling problem is in fact a resource allocation problem, where the resource to be allocated is the energy consumption and/or production of the device. Resource allocation problems are well studied in the literature and have applications in many fields; see for example the excellent surveys (Patriksson 2008; Patriksson and Strömberg 2015). One example of an application field is power dispatch, a problem studied in the traditional, centralized energy management field (see, e.g.Van Den Bosch 1985; Van Den Bosch and Lootsma 1987; Kleinmann and Schultz 1990). Another is that of green computing, where processor speed, and thus energy consumption, is dynamically changed to reduce the overall energy requirements. Within this technique of dynamic voltage and frequency scaling (DVFS) (see, e.g. Yao et al. 1995; Li et al. 2006), problems similar to those we encounter in decentralized energy management are commonly solved (Huang and Wang 2009; Tang et al. 2014). Another area of application is logistics (see, e.g. Norstad et al. 2011; Hvattum et al. 2013). This problem consists of finding optimal vessel route and speed schedules. Herein a resource allocation problem is used as a subroutine of the main solution method. Hence, also in this field, efficient solution methods are highly desirable.

In the area of decentralized energy management, one of the important device scheduling problems is that of finding optimal schedules for the charging of EVs. We show that this problem corresponds to the classical resource allocation problem, as surveyed by Patriksson. For this problem, many (efficient) solution methods exist. Next, we extend the EV problem to include the option to discharge electricity to the grid (vehicle to grid). In this way, the EV can function as a producer during times when this is beneficial. The resulting problem also models the behaviour of various other devices, e.g. a heat pump combined with a heat storage. We derive a property of any optimal solution to this more general problem and use this property to derive an efficient, recursive algorithm, which is based on a divide-and-conquer strategy. The resulting algorithm is also applicable to the aforementioned applications in DVFS and for the vessel route and speed scheduling problem.

In the final part of the paper, we consider variants of the previously studied problems where the decision variables no longer have a continuous range but are restricted to a discrete set, to more closely model the behaviour of several devices occurring in practice. We show these problems are NP-hard in general. However, we derive efficient algorithms for natural relaxations of these problems. Furthermore, we show that these algorithms produce solutions that closely resemble feasible solutions to the original, hard problems.

The major contributions of this work are:

- An efficient solution method for an extension to the traditional resource allocation problem with applications in decentralized energy management and various other fields.
- An NP-hardness proof for the discrete variants of both the classical resource allocation problem and the extension discussed in this work.
- Efficient algorithms for a natural relaxation of both discrete problems considered.

The remainder of this work is organized as follows. In the next section, we introduce the problem of scheduling the charging of an EV and show that this is a continuous resource allocation problem. We then state optimal and necessary conditions from the literature which we require later on. In Sect. 3, we extend the EV charging problem with cumulative intermediate bounds, which follow from the DEM setting. Furthermore, we show that the problem can be solved efficiently and in a recursive manner by solving variants of the original EV problem. In Sect. 4, we consider the case where the decision variables are restricted to a finite set of possible options, showing NP-hardness and considering solution methods to natural relaxations for both problems discussed previously. Finally, some conclusions are drawn in Sect. 5.

2 Background and definition of the EV charging problem

In this section, we consider the problem of scheduling the energy consumption (i.e. the load profile) of an EV under a received steering signal. We assume that the schedule is made for a time horizon discretized into a set of time intervals, i.e. the schedule details how much energy is charged for every time interval of, for example, 15 min. Let i denote the index of the time intervals, running from 1 to n, and x_i be the scheduled energy consumption of the EV for time interval i. We assume that the steering signals

induce a cost function $f_i(x_i)$ for every time interval i. Examples of cost functions are the price of the consumed energy ($f_i(x_i) := a_i x_i$ with a_i the price of energy during interval i) or the squared deviation from a target consumption ($f_i(x_i) = (x_i - d_i)^2$ with d_i the desired consumption for interval i). The total cost of a schedule is given by the sum $\sum_{i=1}^{n} f_i(x_i)$ of the costs for the individual intervals. Note that by this cost definition we implicitly assume that the total cost is separable. Furthermore, we assume that the cost for an interval, given by f_i, is convex and continuous in x_i.

Next to the costs, also some constraints influence the schedule for the EV's load profile. The EV is assumed to arrive at a known point in time and has to be fully charged before its next departure, which is assumed to be known. Formally, this means that for the EV controller we assume the EV arrives at the beginning of time interval 1 and leaves at the end of time interval n. The total required charging is also assumed to be known and is given by C. This implies that $\sum_{i=1}^{n} x_i = C$ has to be fulfilled to ensure that the EV is fully charged before departure. Finally, we do not allow the vehicle to discharge energy, i.e. we require that $x_i \geq 0$, and we limit the total charging done on time interval i by u_i, a given parameter that results from, for example, the maximal allowed charging of the battery of the EV, i.e. $x_i \leq u_i$. This leads to the following optimization problem:

$$\begin{aligned} \min_x \ & f(x) = \sum_{i=1}^{n} f_i(x_i), && (SRA) \\ \text{s.t.} \ & \sum_{i=1}^{n} x_i = C, \\ & 0 \leq x_i \leq u_i \qquad i = 1, 2, \ldots, n. \end{aligned}$$

This problem is known in the literature as nonlinear continuous resource allocation and is extensively studied in many fields of application (see, e.g. Hochbaum and Hong 1995; Bretthauer and Shetty 2002). Note that we can transform the case where also a lower bound $l_i \geq 0$ on the charging done in interval i is given, i.e. $l_i \leq x_i \leq u_i$, to problem *SRA* by the variable substitution $x_i' = x_i - l_i$.

Many efficient methods to solve problem *SRA* exist (see, e.g. Patriksson 2008; Patriksson and Strömberg 2015). Most of these methods are based on optimality conditions tracing back to the nineteenth-century scientist Gibbs (as noted by Patriksson 2008). These conditions were also used in previous work in the field of energy management to obtain solution methods to similar problems (Van Den Bosch 1985; Van Den Bosch and Lootsma 1987; Kleinmann and Schultz 1990). The optimality conditions can be shown to be necessary and sufficient using a generalization of the well-known KKT conditions (Boyd and Vandenberghe 2004). We state them in the following lemma.

Lemma 1 (Necessary and sufficient optimality conditions for *SRA*) *A solution x to SRA is optimal if and only if there exists a multiplier λ such that:*

$$0 < x_i < u_i \Rightarrow \quad f_i^-(x_i) \leq \lambda \leq f_i^+(x_i),$$

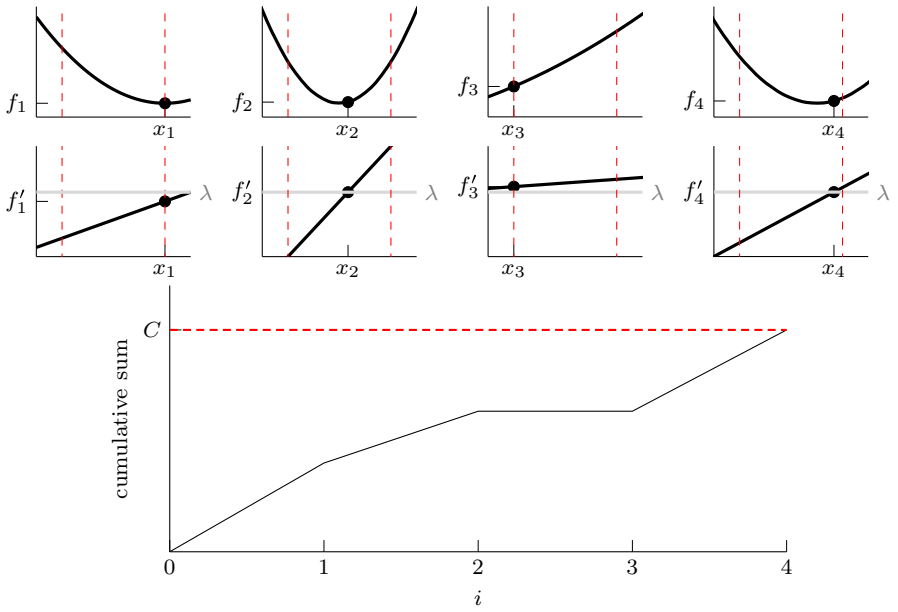


Fig. 1 Example of problem *SRA* as described in Example 1

$$x_i = 0 \Rightarrow \quad f_i^+(x_i) \geq \lambda,$$
$$x_i = u_i \Rightarrow \quad f_i^-(x_i) \leq \lambda.$$

where f_i^- and f_i^+ denote the left and right derivatives of f_i, respectively.

Proof follows directly from the generalized KKT conditions with subdifferentials (see, e.g. Rockafellar 1970). □

These optimality conditions play a vital role when considering extensions of problem *SRA*. Furthermore, we note that it is possible to explicitly formulate the solution from the conditions in case the objective functions are quadratic (see Kleinmann and Schultz 1990). Example 1 gives some more insight in the conditions.

Example 1 Figure 1 depicts an example solution to an instance of Problem *SRA* with $n = 4$. In the first four plots of the figure, we plot the 4 objective functions $f_1, \ldots, f_4$ with the optimal solution $x = (x_1, \ldots, x_4)$. By Lemma 1, there exists a λ such that $f_i'(x_i) = \lambda$ for i with $0 < x_i < u_i$. This is the case for $i = 2$ and $i = 4$ in our example. Furthermore, $f_1'(x_1) < \lambda$ and $f_3'(x_3) > \lambda$ since $x_1 = 0$ and $x_4 = u_4$. The derivatives for this particular instance are given in the next 4 plots with the value of λ given by the grey line. In the last plot, the cumulative sum $\sum_{i'=1}^{i} x_{i'}$ is shown. We note that the only restriction on this sum is that it should be equal to C for $i = 4$.

3 Problem *SRA* with cumulative bounds

In the previous section, we considered the problem of deriving a charging schedule for an EV under various steering signals. We noted that it is in fact a resource allocation problem for which efficient solution methods exist. We assumed that the EV only charges energy from the grid. However, as the battery used to power an EV is typically much larger than the average daily energy requirement for travel, the battery can also supply energy back to the grid when this is beneficial, e.g. during periods of excessive demand or in case of contingencies in the main electricity grid. This can be incorporated in the model by allowing the values of x_i to also take negative values, i.e. by replacing the constraint $x_i \geq 0$ by $x_i \geq l_i$ for given $l_i < 0$. Note that in this case we must take care that the state of charge of the battery does not exceed its limits. To model this, we replace the single resource constraint in problem *SRA* by a cumulative bound on the resource usage for every time interval:

$$\begin{aligned} \min_x \ & f(x) = \sum_{i=1}^{n} f_i(x_i), && (CRA) \\ \text{s.t. } & B_j \leq \sum_{i=1}^{j} x_i \leq C_j \quad j = 1, 2, \ldots, n, \\ & 0 \leq x_i \leq u_i \quad i = 1, 2, \ldots, n. \end{aligned}$$

Note that we applied the transformation $x'_i = x_i - l_i$ to reformulate the problem to include a non-negativity constraint on x_i instead of a more general lower bound. Furthermore, note that problem *CRA* is similar to problem *Nested* discussed by Hochbaum and Hong (1995). However, our formulation is more general, as we allow for *both* an upper and a lower bound on the cumulative use of the resource, whereas Hochbaum and Hong (1995) considers only an upper bound.

Problem *CRA* allows modelling of the charging and discharging of an electric vehicle. Furthermore, the same formulation also models a stand-alone battery used inside a house to, for example, store solar energy for later use or a heat producer (e.g. a heat pump or combined heat and power unit) together with a heat vessel to store the heat for usage later on. In the latter case, B_j represents the cumulative heat demand up to time j, which must be met by the device, and C_j represents this demand plus the storage capacity.

Problem *CRA* has many applications outside of decentralized energy management. For example, it is a subroutine in the vessel routing and scheduling problem where an optimal route for a fleet of vessels has to be determined together with the speed for the vessels on the legs of their route (Norstad et al. 2011; Hvattum et al. 2013). Also, problem *CRA* models the problem of finding a trade-off between processor speed and energy consumption and determining a schedule of tasks for such a processor that satisfies arrivals and deadlines while minimizing energy consumption in the field of DVFS with agreeable deadlines (Huang and Wang 2009). Commonly the assumption that the objective functions are equal for all times intervals is made in both aforementioned fields. We note that we do not make this assumption.

When considering problem *CRA*, we note that we no longer have the constraint that a specific amount of the resource must be used over the given set of intervals, i.e. we can have that $B_n < C_n$. However, if this is the case, we can add an additional variable x_{n+1} with $f_{n+1}(x_{n+1}) = 0$, $B_{n+1} = C_{n+1} = C_n$ and $l_{n+1} = 0, u_{n+1} = C_n - B_n$. This essentially allows us to ensure that the total usage of the resource over the set of intervals sums up to C_n without incurring additional cost, hence preserving optimality. An optimal solution to the original problem now follows by discarding the variable x_{n+1}. Hence, in the following, we assume that $B_n = C_n$.

The base of our solution approach is that we first generate a candidate solution by solving problem *SRA* with $C = C_n$ and the given bounds on the resource usage in the different intervals. This solution may be infeasible for several of the cumulative resource constraints, i.e. $\sum_{i=1}^{j} x_i$ might be smaller than B_j or larger than C_j for several j. However, since the objective functions are convex, intuitively it seems that in an optimal solution the constraint that is violated most in the (infeasible) candidate solution should be tight in an optimal solution to problem *CRA*. This property is in fact proven and used in both the applications of DVFS (Huang and Wang 2009) and vessel speed optimization (Hvattum et al. 2013) to construct efficient algorithms to compute a solution. However, in those settings, the objective function is the same for every time interval, whereas in decentralized energy management this does not need to be the case. In the following, we show that this property still holds when we assume each f_i to be an arbitrary continuous and convex function.

Lemma 2 *Consider an instance of CRA with $C_n = B_n$ and let y be an optimal solution to the instance of SRA obtained by ignoring the cumulative bounds for all indices except the last. Assume that y is not feasible for the considered instance of CRA and let k be the index of the variable that maximally violates the cumulative bounds, i.e. $k = \arg\max_j \{\sum_{i=1}^{j} y_i - C_j, B_j - \sum_{i=1}^{j} y_i\}$. Then, there is an optimal solution x to the considered instance of CRA such that, if $\sum_{i=1}^{k} y_i > C_k$, then $\sum_{i=1}^{k} x_i = C_k$ and, on the other hand, if $\sum_{i=1}^{k} y_i < B_k$, then $\sum_{i=1}^{k} x_i = B_k$.*

Proof Let x be an optimal solution to the considered instance of *CRA* and assume that $\sum_{i=1}^{k} y_i > C_k$ and $\sum_{i=1}^{k} x_i \neq C_k$. Since x is feasible, it follows that $\sum_{i=1}^{k} x_i < C_k$. Let l be the last index before k for which the upper cumulative bound is met by x with equality, i.e. $l := \max\{j < k | \sum_{i=1}^{j} x_i = C_j\}$ and $l := 0$ if this set is empty. Furthermore, let m be the first index after k for which the upper cumulative bound is met by x with equality, i.e. $m := \min\{j > k | \sum_{i=1}^{j} x_i = C_j\}$. Note that $\sum_{i=1}^{n} x_i = C_n$ by assumption; hence, m is well defined and $m \leq n$.

Since $\sum_{i=1}^{l} x_i = C_l$ and $\sum_{i=1}^{k} x_i < C_k$, it follows that $\sum_{i=l+1}^{k} x_i < C_k - C_l$. Also, since $\sum_{i=1}^{l} y_i - C_l \leq \sum_{i=1}^{k} y_i - C_k$ by construction of k, it follows that $\sum_{i=l+1}^{k} y_i \geq C_k - C_l$. Combining this, we obtain that $\sum_{i=l+1}^{k} y_i > \sum_{i=l+1}^{k} x_i$, and hence, there exists an index s with $l < s \leq k$ such that $y_s > x_s$. Similarly, since $\sum_{i=1}^{m} x_i = C_m$ and $\sum_{i=1}^{k} x_i < C_k$, it follows that $\sum_{i=k+1}^{m} x_i > C_m - C_k$. Also, since $\sum_{i=1}^{m} y_i - C_m \leq \sum_{i=1}^{k} y_i - C_k$, it follows that $\sum_{i=k+1}^{m} y_i \leq C_m - C_k$. Combining this, we obtain that $\sum_{i=k+1}^{m} y_i < \sum_{i=k+1}^{m} x_i$, and hence, there exists an index t with $k < t \leq m$ such that $y_t < x_t$.

From Lemma 1, we obtain that $f_s^-(y_s) \leq f_t^+(y_t)$. Furthermore, by the convexity of f_s and f_t, respectively, it follows that $f_s^+(x_s) \leq f_s^-(y_s)$ and $f_t^+(y_t) \leq f_t^-(x_t)$. Thus we obtain:

$$f_s^+(x_s) \leq f_s^-(y_s) \leq f_t^+(y_t) \leq f_t^-(x_t) \tag{1}$$

Note that, for $l < j < m$, $\sum_{i=1}^{j} x_i < C_j$. Since $l < s \leq k < t \leq m$ taking $x_s = x_s + \epsilon$ and $x_t = x_t - \epsilon$ does not violate feasibility, furthermore (1) implies that this does not increase the objective value for sufficiently small ϵ. This increases $\sum_{i=1}^{k} x_i$ and we can repeat this process until $\sum_{i=1}^{k} x_i = C_k$. This shows that $\sum_{i=1}^{k} y_i > C_k$ implies that $\sum_{i=1}^{k} x_i = C_k$.

The proof for the case that $\sum_{i=1}^{k} y_i < B_k$ and $\sum_{i=1}^{k} x_i > B_k$ is symmetric. □

Lemma 2 is the base of a solution approach for *CRA*. For this approach, we first ignore the intermediate cumulative bounds of *CRA*, and afterwards, we iteratively satisfy the ignored constraints using a divide-and-conquer approach. We start by calculating an optimal solution for the instance of *SRA*, which we get by setting $C = C_n$. For this optimal solution, we determine the index k where this solution maximally violates the cumulative bounds. By Lemma 2 we know that there exists an optimal solution for which the corresponding bound is tight for index k, meaning that we can set both B_k and C_k to the value of the violated bound (i.e. either to B_k or to C_k). Note that this splits the original instance of *CRA* into two independent instances of *CRA*: one for the indices up to and including k and one for the indices after k. These problems can be solved separately following the same procedure. Hence we can recursively solve problems of the form *SRA*, until we no longer have violations of the intermediate cumulative bounds. Combining the individual solutions of these instances then gives a solution to *CRA* that is optimal by Lemma 2 and the fact that the individual solutions are optimal for their respective time intervals.

This sketched procedure is summarized in Algorithm 1. In this algorithm, we use $f_{i\to j}$ to denote the vector $(f_i, f_{i+1}, \ldots, f_j)$ of objective functions and a similar notation for the parameters u_i, B_i, and C_i and decision variables x_i. Furthermore, $optSRA(f_{1\to n}, u_{1\to n}, C)$ denotes a call to an algorithm that solves an instance of *SRA* with objective functions $f_{1\to n}$ and parameters $u_{1\to n}$ and C [(for this, one can use, for example, the approaches given by Hochbaum and Hong (1995) or Kleinmann and Schultz (1990)]. Such an algorithm outputs a solution vector $x_{1\to n}$ that is optimal for this instance. Example 2 gives some insight into the structural properties of the algorithm.

Example 2 Figure 2 depicts an application of Algorithm 1. The problem instance is the same as in Example 1 except that we added lower and upper cumulative bounds. These bounds are depicted in middle plot showing the cumulative sum. Above the cumulative sum, the objective functions and derivates are plotted together with the original solution x to Problem *SRA*. This solution serves as a candidate solution to Problem *CRA*, which does not consider the cumulative bounds. This candidate solution is not feasible, since $\sum_{i=1}^{2} x_i > C_2$. Based on this, we split the problem into two subproblems. In the first subproblem, we have to decrease x_1 and/or x_2. Note that by doing this we obtain $x_1' < u_1$, and thus, we find λ_1 with $f_1'(x_1') = f_2'(x_2') = \lambda_1$. In the

Algorithm 1 Recursive algorithm *optCRA* for problem *CRA*

$x_{1\to n}$ = **Function** $optCRA(f_{1\to n}, u_{1\to n}, B_{1\to n}, C_{1\to n})$.
$y_{1\to n} = optSRA(f_{1\to n}, u_{1\to n}, C_n)$.
if $y_{1\to n}$ is feasible **then**
  $x_{1\to n} = y_{1\to n}$.
else
  $k = \arg\max\{\sum_{i=1}^{k} y_i - C_k, B_k - \sum_{i=1}^{k} y_i\}$.
  if $\sum_{i=1}^{k} y_i > C_k$ **then**
    $B_k = C_k$
    $B_i = B_i - C_k$ and $C_i = C_i - C_k$ for $i = k+1, k+2, \ldots, n$.
  else
    $C_k = B_k$
    $B_i = B_i - B_k$ and $C_i = C_i - B_k$ for $i = k+1, k+2, \ldots, n$.
  end if
  $x_{1\to k} = optCRA(f_{1\to k}, u_{1\to k}, B_{1\to k}, C_{1\to k})$.
  $x_{k+1\to n} = optCRA(f_{k+1\to n}, u_{k+1\to n}, B_{k+1\to n}, C_{k+1\to n})$.
end if
Return $x_{1\to n}$.

second subproblem, we increase x_3 and x_4. Doing this gives us $x'_4 = u_4$; thus, we find λ_2 such that $f'_3(x'_3) = \lambda_2 > f'_4(x'_4)$. Combining the solutions to the subproblems, we obtain the optimal solution x' to Problem *CRA*. This solution is given in the bottom eight plots.

It remains to determine the complexity of Algorithm 1. Let $F_{SRA}(n)$ denote the complexity of the algorithm $optSRA(f_{1\to n}, u_{1\to n}, C_n)$, called by *optCRA* to solve an instance of *SRA* with n variables. Furthermore, let $F_{CRA}(n)$ be the complexity of Algorithm 1 for instances with n variables. We then obtain the following recursive relation for $F_{CRA}(n)$:

$$F_{CRA}(n) = O(n) + F_{SRA}(n) + F_{CRA}(k) + F_{CRA}(n-k) \tag{2}$$

Assuming that $F_{SRA}(n)$ has a complexity of $\Omega(n)$, we have that $F_{SRA}(k) + F_{SRA}(n-k) \leq F_{SRA}(n)$. This implies that $F_{CRA}(n) = O(n^2 + nF_{SRA}(n))$. We note that, for the case that the objective functions are quadratic, Hochbaum and Hong (1995) provide an $O(n)$ algorithm to solve Problem *SRA*. Combining this with our approach yields a complexity of $O(n^2)$. Furthermore, we note that it is also possible to combine our approach with one of the approaches classically used in energy management to solve Problem *SRA*.

4 Resource allocation over a discrete set

In the previous sections, we assumed that the scheduled load of the device in an interval can be any value between the given lower and upper bound for that interval. However, this might not be the case in practice for several devices. For example, a heat pump generally runs at a limited number of predefined levels. Also, current charging stations

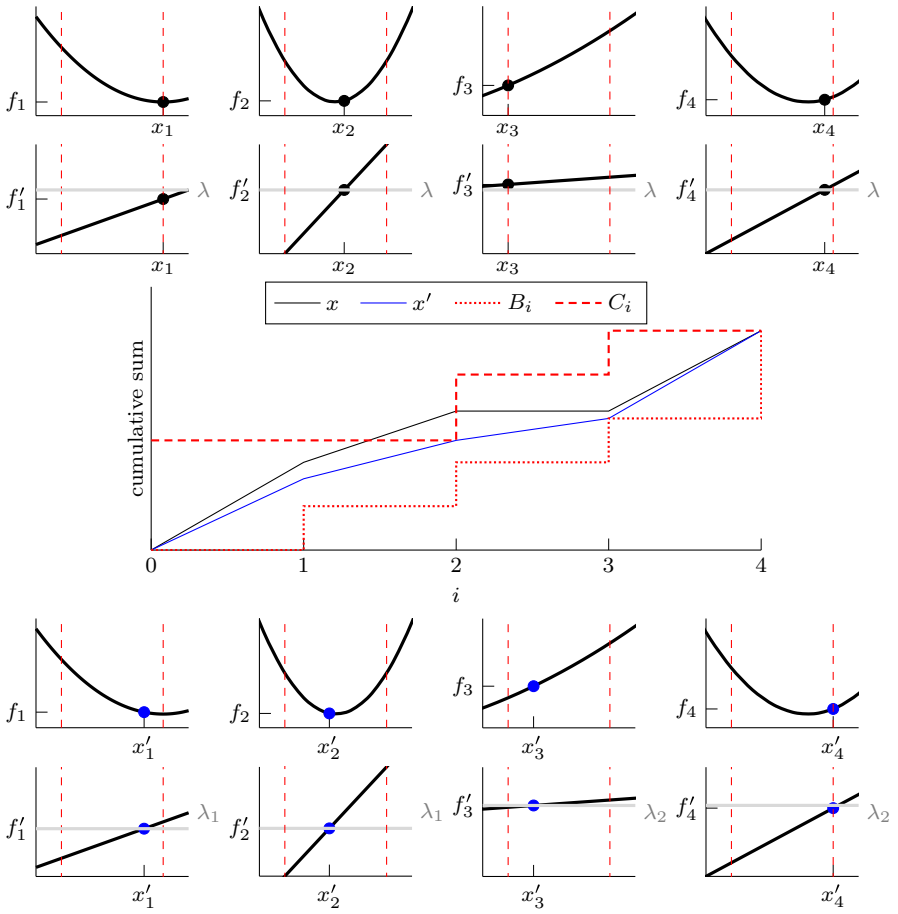


Fig. 2 Example of an application of Algorithm 1 as described in Example 2

for EVs only allow charging at specific amperages, i.e. at specific power levels (Kesler et al. 2014). To model this, we have to replace the continuous range for x_i by a finite set. We show that this makes problem *SRA* NP-hard in general. However, we show that there is a natural relaxation in the setting of decentralized energy management that leads to efficient solution methods which give solutions that are quite close and similar to solutions to the original discrete version of the problem.

4.1 Discrete *SRA*

In this section we consider the discrete variant of Problem *SRA*. In this variant, we replace the constraint $0 \leq x_i \leq u_i$ in *SRA* by $x_i \in Z_i := \{z_i^0, z_i^1, \ldots, z_i^{m_i}\}$. We note that we can safely assume that $z_i^0 = 0$, by applying the transformation $x_i' = x_i - z_i^0$. Furthermore, we do not assume that the sets Z_i are equal for all time interval. Finally, we note that the problem is different from discrete resource allocation considered in the

literature (see, e.g. Hochbaum and Hong 1995), because in the literature the standard assumption is that $Z_i = \mathbb{Z}$ for each i. This means that the discrete variant occurring in the DEM context is more general. Formally, the discrete version of problem *SRA* is given as:

$$\begin{aligned} &\min_x \sum_{i=1}^{n} f_i(x_i), && (dSRA)\\ &\text{s.t.} \sum_{i=1}^{n} x_i = C,\\ &x_i \in Z_i \quad i = 1, 2, \ldots, n. \end{aligned}$$

Contrary to the discrete problems found in the literature, this problem is NP-hard, as shown by a reduction from the partition problem. This result still holds when the sets Z_i are the same for all i, as we show below.

Lemma 3 *The decision problem of determining whether a feasible solution to dSRA exists is NP-complete, even if all sets Z_i are the same.*

Proof Clearly, the problem of existence of a feasible solution is in NP. The NP-completeness if the sets Z_i may differ follows from a reduction from the partition problem. For this we define $Z_i := \{0, p_i\}$ and $C = \frac{1}{2}\sum_i p_i$, where $p_1, p_2, \ldots, p_n$ are the integers from the set to be partitioned.

For the case that all Z_i are equal, we assume that $Z_i = \{z_0, z_1, \ldots, z_m\}$ for every i and use a reduction from even/odd partition. In the even/odd partition problem, a set $P = \{p_1, p_2, \ldots, p_{2l}\}$ of $2l$ non-negative integers is given with total sum $2B$. The problem asks whether there is a subset $A \subset \{1, 2, \ldots, 2l\}$ such that $\sum_{i\in A} p_i = B$ and for $i = 1, 2, \ldots, l$ we have: $2i - 1 \in A \Leftrightarrow 2i \notin A$.

Consider an instance I of even/odd partition and let $k := \lfloor \log_2(2B) \rfloor$, i.e. k is the unique integer such that $2^k \leq 2B < 2^{k+1}$. To transform this instance of even/odd partition to an instance I' of *dSRA*, we take $n = m = 2l$. Furthermore, we choose $z_{2i-1} = p_{2i-1} + 2^{k+i}$ and $z_{2i} = p_{2i} + 2^{k+i}$ for $i = 1, 2, \ldots, l$ and $C = B + \sum_{i=1}^{l} 2^{k+i}$. In the following, we show that I is a yes-instance iff I' is a yes-instance.

First assume that I is a yes-instance. Thus there exists a subset A with $\sum_{i\in A} p_i = B$ and $2i \in A \Leftrightarrow 2i - 1 \notin A$. By defining $x_{2i} = p_{2i} + 2^{k+i}$ if $2i \in A$ and $x_{2i} = 0$ otherwise, and $x_{2i-1} = p_{2i-1} + 2^{k+i}$ if $2i - 1 \in A$ and $x_{2i-1} = 0$ otherwise, it now follows that $\sum_{i=1}^{n} x_i = \sum_{i\in A} p_i + \sum_{i=1}^{l} 2^{k+i} = C$.

On the other hand, assume that I' is a yes-instance for the *dSRA* problem and x a corresponding solution. Note that the l most significant bits of C in binary representation are 1 and correspond to $2^{k+1}, 2^{k+2}, \ldots, 2^{k+l}$. Since $\sum_{t=1}^{n} x_t = C$, it follows that for $i = 1, 2, \ldots, l$ there is exactly one of the values $p_{2i} + 2^{k+i}$ or $p_{2i-1} + 2^{k+i}$ contained in $\sum_{i=1}^{n} x_i$. If we now define A as the subset of $\{1, 2, \ldots, n\}$ of the indices of these values that are contained in $\sum_{i=1}^{n} x_i$, it follows that $2i \in A \Leftrightarrow 2i - 1 \notin A$. Furthermore, by construction we have $\sum_{i\in A} p_i = \sum_{t=1}^{n} x_t - \sum_{t=1}^{l} 2^{k+t} = B$.

This shows that checking whether a feasible solution to *dSRA* exists is NP-complete, even if we assume that the Z_i are the same for every index i. □

The above lemma shows that checking whether a feasible solution exists for *dSRA* is NP-complete. However, in the case of energy management, the variable x_i expresses the power produced or consumed by the device over a time interval. Restricting this value to be constant over an entire time interval does not reflect all operational possibilities as many flexible devices have the option to shift between different power levels at any moment. We may model this by modifying problem *dSRA* to allow convex combinations of the points in each set Z_i, i.e. we introduce multipliers $y_i^0, y_i^1, \ldots, y_i^{m_i} \in [0, 1]$ with $\sum_j y_i^j = 1$ and $\sum_j y_i^j z_i^j = x_i$. We note that this is also common practice in DVFS (see, e.g. Kwon and Kim 2005; Li et al. 2006).

Allowing convex combinations within an interval leads to the question how the objective value for interval i, denoted by F_i, should be defined. There are two options, either we use the original objective value and compute its value at the convex combination, i.e. $F_i = f_i(\sum_j y_i^j z_i^j)$, or we use the convex combination of the value of the objective function at the points in the set, i.e. $F_i(x_i) := \sum_j y_i^j f_i(z_i^j)$. Note that the former choice essentially transforms the problem back into the continuous version of *SRA*, as any solution to this version of *SRA* can readily be transformed to a solution of the obtained problem with the same objective value by picking the right multipliers. When considering stress put on the electricity network by the load of various devices, the latter objective is preferred, as a switch between a higher and lower load during an interval puts considerably more stress on the network than the average load of the two. In the latter case, the problem becomes:

$$\min_y \sum_{i=1}^{n} F_i(x_i) = \sum_{i=1}^{n} \sum_{j=0}^{m_i} y_i^j f_i\left(z_i^j\right), \tag{rdSRA}$$

$$\text{s.t.} \sum_{i=1}^{n} x_i = C,$$

$$x_i = \sum_{j=0}^{m_i} y_i^j z_i^j \quad i = 1, 2, \ldots, n,$$

$$\sum_{j=0}^{m_i} y_i^j = 1 \quad i = 1, 2, \ldots, n,$$

$$y_i^j \geq 0 \quad i = 1, 2, \ldots, n;\ j = 0, 1, \ldots, m_i.$$

Considering problem *rdSRA*, we note that, by the convexity of f_i, we can assume that, for any i, only at most two consecutive multipliers y_i^j, y_i^{j+1} are nonzero. Hence we can replace F_i by the piecewise linear function obtained from f_i by linearizing it on each of the intervals $(z_i^0, z_i^1), (z_i^1, z_i^2), \ldots, (z_i^{m_i-1}, z_i^{m_i})$. By doing this we obtain an instance of problem *SRA* with piecewise linear objective functions.

As mentioned before, we are interested in very fast and efficient solution methods, since the considered problems generally have to be solved often and on embedded platforms. To this end, we study Problem *SRA* with piecewise linear objective in more

detail. The breakpoints of the piecewise linear function f_i are given by the set Z_i. Furthermore, we use s_i^j to denote the slope of the piece between breakpoints z_i^{j-1} and z_i^j. We first observe that pieces with a smaller slope will always be preferred by an optimal solution.

Lemma 4 *Consider an instance of Problem SRA where each f_i is a piecewise linear function. Furthermore, assume that a piece with slope s is used in an optimal solution (i.e. the corresponding x_i is larger than the left breakpoint of this piece). Then, any piece with a slope smaller than s is also used by the optimal solution.*

Proof Consider an optimal solution x and assume there is a piece with slope $s' < s$, which is not used, with corresponding objective function $f_{i'}$. Let $s^\star$ be the slope of the first piece of $f_{i'}$ that is not completely used by x. By the convexity of $f_{i'}$, it follows that $s^\star \leq s' < s$. This is a contradiction with Lemma 1. □

From Lemma 4 it follows that a greedy approach preferring pieces of functions with smaller slopes is optimal. Thus, calculating an optimal solution to Problem *SRA* with piecewise linear objective comes down to sorting the pieces such that their slopes are non-decreasing and then using these pieces until $\sum_i x_i = C$. Note that the pieces of each objective function f_i are automatically correctly ordered due to the convexity of the f_i. A straightforward implementation considers all pieces simultaneously and sorts them based on their slopes, resulting in a complexity of $O(M \log M)$, where M is the sum of the number of pieces m_i of all n objective functions. However, we can reduce the complexity to $O(M \log n)$ by only considering a single piece per objective function at a time. The resulting greedy approach is summarized below in Algorithm 2. We use the same notation for the objective functions, decision variables and parameters as in Algorithm 1. Furthermore, we use Z to denote the set containing all the breakpoints of all functions f_i and s_i^j to denote the slopes of the linear pieces of f_i between breakpoints z_i^{j-1} and z_i^j. Example 3 illustrates the working of Algorithm 2.

Example 3 Figure 3 depicts an application of Algorithm 2. In this example, piecewise linear approximations of the objective functions used in Examples 1 and 2 are used. These approximations are given in the upper four plots with the derivatives, i.e. the slopes of the pieces, given in the four plots below that and the cumulative sum given in the last plot. The particular solution depicted in the plots is the solution after 2 steps in the while loop of the algorithm, where the first piece of both f_2 and f_4 (both in blue) has already been used. In the next step, the first piece of f_1 (red dotted) is used since it has the lowest slope, i.e. it is the first slope in S. After this, the second piece of f_1 (green dashed) is added. The ordered set S over the different steps of the algorithm is given below the plots in the lower part of the figure, with the piece used and removed marked in red and crossed out (note that this is always the first piece of S) and the new piece added marked in green. Note that in the last step the piece s_2^2 is only partially used.

Lemma 5 *Algorithm 2 solves SRA with piecewise linear convex objective functions to optimality in $O(M \log n)$ time, where M is the sum of the number of pieces of the objective functions.*

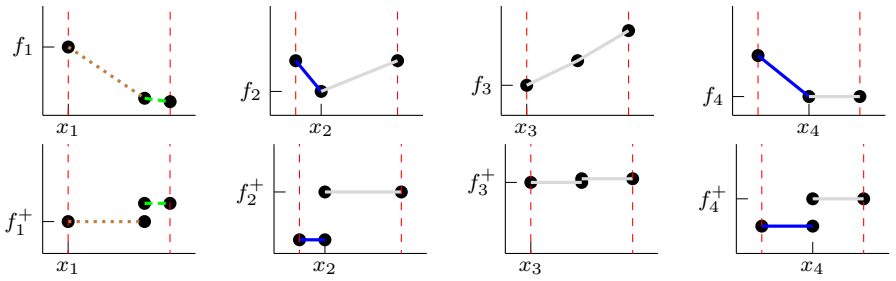
Algorithm 2 Greedy approach *pwlSRA* for SRA with piecewise linear objective

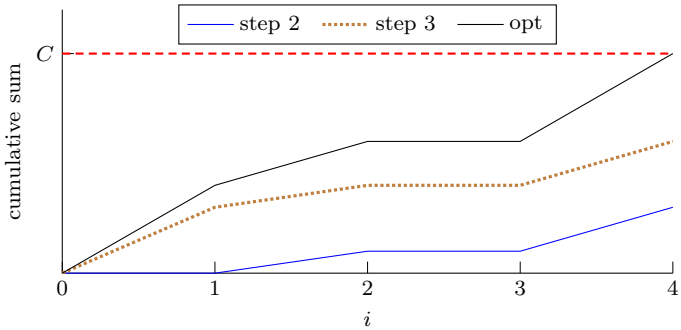
1: $x_{1\to n}$ = **Function** $pwlSRA(f_{1\to n}, Z, C)$.

2: Take $S = \{s_1^1, s_2^1, \ldots, s_n^1\}$ and order S non-decreasingly.

3: Set $x_i = 0$ for all i.

4: **while** $C > 0$ **do**
5: Take the first slope s_i^j from S which belongs to a piece with associated i and j.
6: $\delta := \min\{C, z_i^j - z_i^{j-1}\}$.
7: $x_i = x_i + \delta$
8: $C = C - \delta$.
9: $S = S \setminus \{s_i^j\}$
10: **if** $j < m_i$ **then**
11: Insert s_i^{j+1} into the ordered set S.
12: **end if**
13: **end while**
14: Return $x_{1\to n}$.





Step	S
Initial	$\{s_2^1, s_4^1, s_1^1, s_3^1\}$
1	$\{\cancel{s_2^1}, s_4^1, s_1^1, s_2^2, s_3^1\}$
2	$\{\cancel{s_4^1}, s_1^1, s_4^2, s_2^2, s_3^1\}$
3	$\{\cancel{s_1^1}, s_1^2, s_4^2, s_2^2, s_3^1\}$
4	$\{\cancel{s_1^2}, s_4^2, s_2^2, s_3^1\}$
5	$\{\cancel{s_4^2}, s_2^2, s_3^1\}$
6	$\{\cancel{s_2^2}, s_3^1\}$

Fig. 3 Example of an application of Algorithm 2 as described in Example 3

Proof The feasibility of the algorithm follows from the convexity of the f_i's, and the optimality follows directly from Lemma 4.

When considering the time complexity, note that the first sorting in Line 2 can be done in time $O(n \log n)$. Furthermore, the heaviest operation in the while loop is the insertion of the slope of a piece in the ordered vector of at most $n-1$ other slopes. This can be done in time $O(\log n)$ when using an appropriate data structure. Since the while loop runs for at most M iterations and $M \geq n$, it follows that the total complexity is $O(M \log n)$. □

Hochbaum and Hong (1995) exploit the linear complexity of median find to solve problem *SRA* with quadratic objective functions in linear time. Based on their findings, we may formulate an approach for problem *rdSRA* with complexity $O(M)$. Instead of sorting (part of) the pieces such that they have non-decreasing slopes, we iteratively guess the slope of the last piece used by an optimal solution. For this guess, we use the piece with the median value of the slopes, which we can find in linear time (Blum et al. 1973). In case of an even number of slopes, we pick either of the two middle values. After this we split the pieces in sets $S_1 := \{s_i^j | s_i^j \leq m\}$ and $S_2 := \{s_i^j | s_i^j > m\}$, where m is the slope of the found median piece, and determine a solution x with $\hat{C} := \sum_i x_i$ using the pieces in S_1.

Next we consider three cases.

- If $\hat{C} = C$, we have found an optimal solution.
- If $\hat{C} > C$, we check whether we can obtain a feasible, and hence optimal, solution, by using the piece with slope m only partially. If this is not the case, we know that an optimal solution cannot use pieces from S_2 nor piece m. Thus, we recursively call the algorithm using only the pieces in $S_1 \setminus \{m\}$
- If $\hat{C} < C$, we know that an optimal solution must use all the pieces in S_1. Thus, we mark these pieces as used and recursively call the algorithm on the pieces in S_2 with $C := C - \hat{C}$, since the remaining pieces must be used for this amount.

The approach is summarized in Algorithm 3, where we use similar notation to that in Algorithm 2. Furthermore, Example 4 demonstrates an application of the algorithm.

Example 4 Figure 4 depicts an application of Algorithm 3. The same instance is used as in Example 3. Here we depict the first application of the algorithm. Since there are a total of 8 pieces, the algorithm picks $S_1 = \{s_1^1, s_1^2, s_4^1, s_2^1\}$, $S_2 = \{s_2^2, s_3^1, s_3^2, s_4^2\}$, and $m = s_1^2$ in the top plots of the figure in blue, grey and dotted green, respectively. The guessed solution has a cumulative sum $\hat{C}$ lower than C, as shown. For clarity, the cumulative sum of the solution without m is also plotted. This would be used in case $\hat{C} > C$. In the depicted case, the algorithm concludes that each of the currently used pieces, i.e. those in the set S_1, must be used in an optimal solution and it is recursively called on the set S_2 with $C := C - \hat{C}$.

Corollary 1 *Algorithm 3 solves rdSRA to optimality in time $O(M)$, with $M = \sum_i m_i$, i.e. the total number of pieces.*

Algorithm 3 Median find approach *mpwlSRA* for SRA with piecewise linear objective

$x_{1\to n} =$ **Function** *mpwlSRA*$(f_{1\to n}, Z, C)$.
Take S as the set of slopes of all the pieces, $S_1, S_2 := \emptyset$ and $\hat{C} := 0$.
Take $l_i^j := z_i^j - z_i^{j-1}$ for each piece.
Set $x_i = 0$ for all i.
Take m the median of S with associated i_m and j_m.
for i, j **do**
  if $s_i^j > m$ **then**
    $S_2 = S_2 \cup \{s_i^j\}$.
  else
    $S_1 = S_1 \cup \{s_i^j\}$, $\hat{C} = \hat{C} + l_i^j$, and $x_i = x_i + l_i^j$.
  end if
  if $\hat{C} \geq C$ **then**
    if $\hat{C} - l_{i_m}^{j_m} \leq C$ **then**
      $x_{i_m} = x_{i_m} - (\hat{C} - C)$.
    else
      Take $\hat{Z}$ the set of breakpoints for pieces in $S_1 \setminus \{m\}$.
      $x_{1\to n} =$ *mpwlSRA* $(f_{1\to n}, \hat{Z}, C)$.
    end if
  else
    Take $\hat{Z}$ the set of breakpoints for pieces in S_2.
    $\hat{x}_{1\to n} =$ *mpwlSRA* $(f_{1\to n}, \hat{Z}, C - \hat{C})$.
    $x_{1\to n} = x_{1\to n} + \hat{x}_{1\to n}$
  end if
end for
Return $x_{1\to n}$.

Proof The optimality of the algorithm follows immediately from the fact that pieces are used in order; hence, the algorithm finds the same solution as Algorithm 2 which is optimal by Lemma 5. The complexity of $O(M)$ follows from the fact that all the steps before a potential recursive call in the algorithm can be solved in time linear in the number of pieces formed by the breakpoints in Z. For the first call this number of pieces is exactly M, and for subsequent, recursive calls the number is halved each time. Hence the complexity is $O(M + M/2 + M/4 + \cdots) = O(M)$. □

We note that the actual running time of the median find algorithm with linear asymptotic complexity is high for small- to medium-sized sets (Blum et al. 1973). Thus, while Algorithm 3 has a lower asymptotic complexity than Algorithm 2, the latter might in practice be more efficient for decentralized energy management applications, where the number of intervals is typically low.

As mentioned before, for problem *rdSRA*, which we can solve efficiently using either Algorithm 2 or 3, we assumed that devices can switch between different consumption levels or states somewhere during a time interval. However, excessive switching may cause undesirable wearing of the device, e.g. the lifetime reduction that may occur for heat pumps and compressors in air-conditioning systems due to frequent on/off

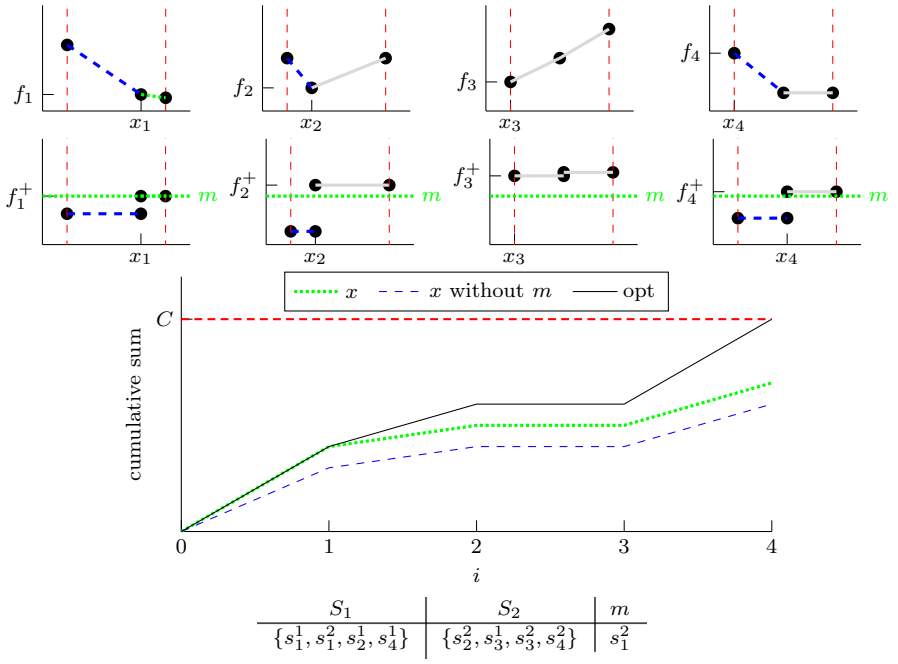


Fig. 4 Example of an application of Algorithm 3 as described in Example 4

cycling or increased degradation of the internal battery within EVs from excessive switching between different charging and discharging currents (Kesler et al. 2014). Thus, solutions that avoid excessive switching are desirable. However, if we look at the solutions produced by Algorithms 2 and 3, we see that they have $x_i \notin Z_i$ for at most one index i. This leads to the following corollary.

Corollary 2 *There always exists an optimal solution to problem rdSRA such that* $x_i \notin Z_i$ *for at most one* i.

As a consequence, the optimal solutions we obtained for Problem *rdSRA* do not cause much extra wearing of the devices over any feasible solution to Problem *dSRA*.

4.2 Discrete *CRA*

In the previous section, we restricted the feasible set of the decision variable x_i for problem *SRA* to the discrete set Z_i for every i. The same restriction can be applied to problem *CRA*. As Problem *CRA* is more general than *SRA*, it readily follows from Lemma 3 that this problem is also NP-hard. Again, as for Problem *SRA*, we now consider the case where we allow convex combinations of the points in Z_i. This leads to the following problem:

$$\min_{y} \sum_{i=1}^{n} F_i(x_i) = \sum_{i=1}^{n} \sum_{j=0}^{m_i} y_i^j f_i\left(z_i^j\right), \qquad (rdCRA)$$

$$\text{s.t. } B_i \le \sum_{i'=1}^{i} x_{i'} \le C_i \quad i = 1, 2, \ldots, n,$$

$$x_i = \sum_{j=0}^{m_i} y_i^j z_i^j \quad i = 1, 2, \ldots, n,$$

$$\sum_{j=0}^{m_i} y_i^j = 1 \quad i = 1, 2, \ldots, n,$$

$$y_i^j \ge 0 \quad i = 1, 2, \ldots, n;\ j = 0, 1, \ldots, m_i,$$

By similar reasoning to that in Sect. 3, we can again assume that $B_n = C_n$. Also, we can consider an instance of *CRA* with piecewise linear objective functions, by similar reasoning to that in Sect. 4.1. Combining Algorithms 1 and 2 gives us an algorithm that runs in time $O(n^2 + nM \log n)$, which is equal to $O(nM \log n)$ since $n \le M$. Furthermore, combining Algorithms 1 and 3 gives an algorithm that runs in time $O(n^2 + nM) = O(nM)$.

Similarly to the greedy approach for Problem *SRA* with piecewise linear objective, presented in Sect. 4.1, we now construct a greedy approach to solve *CRA* with piecewise linear objective that prefers pieces with a smaller slope. However, we also need to satisfy the cumulative bounds B_i and C_i for every i. We note that to satisfy the lower bound B_i, we can only use pieces of the functions $f_{i'}$ with $i' \le i$. Therefore, we iteratively build up a solution that satisfies the lower bounds $B_1, B_2, \ldots, B_k$ in iteration k. To do this, during iteration k, we only consider pieces of the functions $f_1, f_2, \ldots, f_k$ and increase the use of these pieces until $\sum_{i=1}^{k} x_i = B_k$.

Next we consider the upper cumulative bound C_i. We note that we cannot increase the use of the pieces of function f_i by more than $C_j - \sum_{i'=1}^{j} x_{i'}$ for every $j \ge i$, lest we violate the upper bound C_j. Thus, to ensure feasibility, we introduce a variable $V_i := \min_{j \ge i}\{C_j - \sum_{i'=1}^{j} x_{i'}\}$ to track how much we can increase x_i, using the pieces of f_i, without violating the upper bound C_i. Note that V_i depends on all x_j; thus, after we increased the use of the pieces of some f_j by δ, we need to update V_i for every i. Furthermore, note that whenever we increase x_j by δ, $\sum_{i'=1}^{i} x_{i'}$ is also increased by δ for every $i \ge j$. This implies that V_i decreases by exactly δ for every $i \ge j$. On the other hand, for $i < j$, we note that $V_{i-1} = \min\{V_i, C_i - \sum_{i'=1}^{i-1} x_{i'}\}$. This can be used to iteratively update $V_{j-1}, V_{j-2}, \ldots, V_1$.

Finally, to obtain a better complexity, similar to the approach we took in Algorithm 2, we only consider at most one piece per function f_i. This procedure is summarized in Algorithm 4, where we used the same notation as in the algorithms previously presented. Furthermore, Example 5 demonstrates an application of the algorithm.

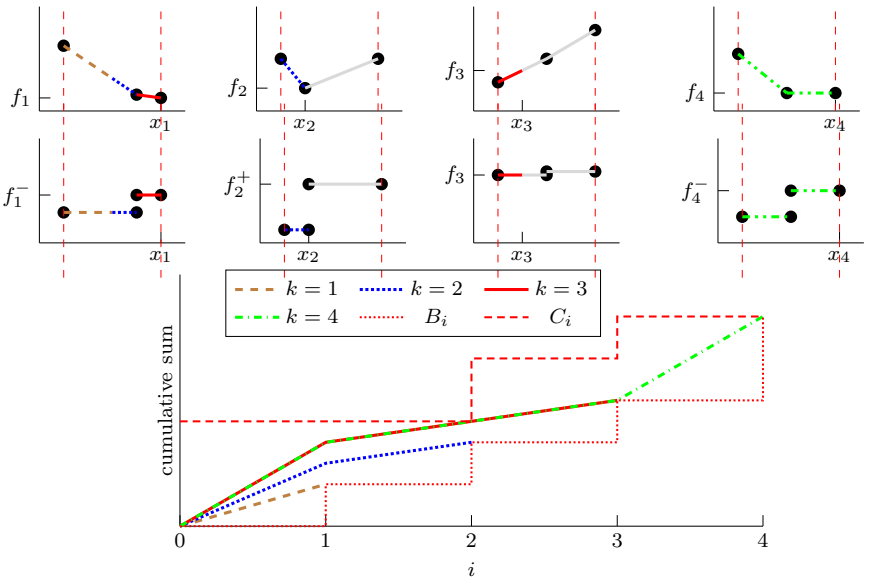
Example 5 Figure 5 depicts an application of Algorithm 4. For this instance, we added the cumulative bounds used in Example 2 to the instance used in Examples 3 and 4. A

Algorithm 4 Greedy approach *pwlCRA* for CRA with piecewise linear convex objective

1: $x_{1\to n}$ = **Function** *pwlCRA*($f_{1\to n}$, Z, $u_{1\to n}$, $B_{1\to n}$, $C_{1\to n}$).
2: $S = \emptyset$, and $V_i = C_i$ and $x_i = 0$ for all i.
3: **for** k=1,2,…,n **do**
4: Insert s_k^1 into S such that S remains ordered non-decreasingly.
5: **while** $\sum_{i'=1}^{k} x_{i'} < B_k$ **do**
6: Take the first piece s_i^j from S
7: $\delta := \min\{R, V_i, z_i^j - z_i^{j-1}\}$.
8: $x_i = x_i + \delta$, $R = R - \delta$.
9: **for** l=i,i+1,…,n **do**
10: $V_l = V_l - \delta$.
11: **end for**
12: $W := V_i$.
13: **for** l=i-1,i-2,…,1 **do**
14: $W = \min\{W, V_l\}$, $V_l = W$.
15: **end for**
16: **for** $s_{i'}^{j'} \in S$ with $V_{i'} = 0$ **do**
17: $S = S \setminus \{s_{i'}^{j'}\}$.
18: **end for**
19: **if** $\delta = z_i^j - z_i^{j-1}$ **then**
20: $S = S \setminus \{s_i^j\}$.
21: **if** $j < m_i$ and $V_i > 0$ **then**
22: Insert s_i^{j+1} into the ordered vector S.
23: **end if**
24: **else if** $V_i > 0$ **then**
25: $z_i^{j-1} = z_i^{j-1} + \delta$.
26: **end if**
27: **end while**
28: **end for**
29: Return $x_{1\to n}$.

colour coding is used to denote (parts of) the pieces that are used in each iteration of the main for loop of the algorithm (brown dashed, blue dotted, red and green dashdotted, respectively). The same colour coding is used in the plot that depicts the cumulative sum till the kth interval for each of the partial solutions constructed. Furthermore, the solution marked in the top plots is the optimal solution the algorithm produces after completion. In the table, the values of various variables used inside the algorithm are denoted for each iteration of the while loop in the algorithm. These iterations are denoted in the column Iteration. Here, S_{start} denotes the set S at the start of the considered iteration of the while loop, and S_{end} the set S at the end of the iteration. New additions to S are depicted in green. In particular we note that s_2^2 is deleted from S at the end of iteration 4, since $V_2 = 0$. Thus the algorithm must use s_3^1 instead to ensure that the bound B_3 is met.

Lemma 6 *The greedy approach for CRA with piecewise linear convex objective, given in Algorithm 4, gives an optimal solution to CRA with piecewise linear convex objective functions. The algorithm runs in time* $O(nM)$.



Iteration	k	S_{start}	S_{end}	(x_1,x_2,x_3,x_4)	sum	B_k	(V_1,V_2,V_3,V_4)
Initial	0	$\emptyset$	$\emptyset$	$(0\,,0\,,0\,,0)$	0	0	$(\frac{5}{2}\,,\frac{5}{2}\,,4\,,5)$
1	1	$\{s_1^1\}$	$\{s_1^1\}$	$(1\,,0\,,0\,,0)$	1	1	$(\frac{3}{2}\,,\frac{3}{2}\,,3\,,4)$
2	2	$\{s_2^1, s_1^1\}$	$\{s_1^1, s_2^2\}$	$(1\,,\frac{1}{2}\,,0\,,0)$	$\frac{3}{2}$	2	$(1\,,1\,,\frac{5}{2}\,,\frac{7}{2})$
3	2	$\{s_1^1, s_2^2\}$	$\{s_1^2, s_2^2\}$	$(\frac{3}{2}\,,\frac{1}{2}\,,0\,,0)$	2	2	$(\frac{1}{2}\,,\frac{1}{2}\,,2\,,3)$
4	3	$\{s_1^2, s_2^2, s_3^1\}$	$\{s_3^1\}$	$(2\,,\frac{1}{2}\,,0\,,0)$	$\frac{5}{2}$	3	$(0\,,0\,,\frac{3}{2}\,,\frac{5}{2})$
5	3	$\{s_3^1\}$	$\{s_3^1\}$	$(2\,,\frac{1}{2}\,,\frac{1}{2}\,,0)$	3	3	$(0\,,0\,,1\,,2)$
6	4	$\{s_4^1, s_3^1\}$	$\{s_4^2, s_3^1\}$	$(2\,,\frac{1}{2}\,,\frac{1}{2}\,,1)$	4	5	$(0\,,0\,,1\,,1)$
7	4	$\{s_4^2, s_3^1\}$	$\{s_3^1\}$	$(2\,,\frac{1}{2}\,,\frac{1}{2}\,,2)$	5	5	$(0\,,0\,,0\,,0)$

Fig. 5 Example of an application of Algorithm 4 as described in Example 5

Proof The feasibility of the algorithm follows from the fact that the pieces are added in increasing order and that any new piece is either the first piece of an objective function (Line 4) or a piece for which the previous piece is already fully used (Line 22). Also, note that the lower cumulative bounds are satisfied since the constructed solution enforces this for every index in the main for loop of the algorithm (Lines 3–28). Finally, note that in every step of the main for loop, $V_i \leq C_j - \sum_{k=1}^{j} x_k$ for $j \geq i$. Hence, in Line 7, we ensure that the upper cumulative bounds are satisfied.

To prove optimality of the algorithm, consider an instance of *CRA* with piecewise linear convex objective functions. Let y be an optimal solution, and let x be the solution produced by Algorithm 4. Furthermore, let i be the smallest index for which $x_i \neq y_i$. We consider two cases.

Case 1 $x_i > y_i$: Let j be the smallest index for which $x_j < y_j$. This index must exist since $\sum_i x_i = B_n = \sum_i y_i$. Furthermore, let s_i and s_j be the slopes of the pieces on which x_i and x_j lie, respectively. In case $x_i \in Z_i$, we pick the piece with x_i as endpoint, and in case $x_j \in Z_j$, we pick the piece with x_j as begin point. By the fact that the right

and left derivatives of f_i and f_j are non-decreasing, it follows that $s_i \geq f_i^+(y_i)$ and $f_j^-(y_j) \geq s_j$. Also, by the optimality of y it follows that $f_i^+(y_i) \geq f_j^-(y_j)$. Finally note that, since both x and y are feasible, we can decrease x_i and increase x_j, until either x_i is equal to y_i or x_j is equal to y_j, without violating feasibility. Finally, note that doing so does not increase the objective value.

Case 2 $x_i < y_i$: This case can be treated completely symmetrical to the previous case.

The above process can be repeated until $x = y$ without increasing the objective value. This shows that x is indeed optimal.

Finally, we show that the time complexity of the algorithm is $O(nM)$ with M the sum of the number of pieces of each f_i, i.e. $M = \sum_i m_i$. Note that a slope s_i^j can only be added to and subsequently removed from S once. Furthermore, if a slope is picked to be used in the while loop and not removed, it follows that $R = 0$ after this iteration, and hence, the while loop is finished. Finally, note that the steps inside the while loop can all be executed in time $O(n)$, since the size of S is never more than n. Hence, the total complexity of the for loop in the algorithm is $O(nM)$, which clearly dominates the complexity of the other steps. □

We note that the asymptotic complexity of Algorithm 4 is slightly lower than the complexity when we recursively apply Algorithm 1 using Algorithm 2 to obtain solutions for the resulting *SRA* problems. However, the latter method might be more favourable in practice, specifically if the number of intervals for which the cumulative bounds are tight is low, as can be expected in many applications (Yao et al. 1995; Vidal et al. 2014). Furthermore, the asymptotic complexity of Algorithm 4 is the same as the complexity when we recursively apply Algorithm 1 using Algorithm 3. As noted before, however, the latter method is probably only efficient for large instances, i.e. instances where the number of breakpoints is very large.

From a practical point of view, it is again important to consider how often solutions to *rdCRA* switch between the different values in the feasible set Z_i for each i. We obtain the following result, which is similar to Corollary 2 for Problem *rdSRA*.

Corollary 3 *There exists an optimal solution to rdCRA such that for any two indices* $i < i'$ *with* $x_i \notin Z_i$, $x_{i'} \notin Z_{i'}$ *there exists an index* $i \leq k < i'$ *with* $\sum_{i''=1}^{k} x_{i''} \in \{B_k, C_k\}$, *i.e. solution* x *meets either the lower or upper cumulative bound tightly for an index in between* i *and* i'.

This result follows immediately from the fact that we can recursively apply Algorithm 2 to Problem *rdCRA* by Lemma 2. In practice we do not expect these cumulative bounds to be met often. Therefore, the above shows that, in most practical cases, the expected number of time intervals for which two operational values are chosen instead of one is low.

5 Conclusion

In this work, we considered scheduling problems motivated by the domain of decentralized energy management. Within this domain, flexible appliances have to schedule

their load profile based on steering signals received from a central controller. We showed that many of these problems can be modelled as resource allocation problems. Furthermore, several variants of these scheduling problems extend the traditional resource allocation problem by adding cumulative lower and upper bounds on the resource usage. This problem has applications in various fields, and we showed that a simple, recursive algorithm can be used to solve this problem in a very general form. The asymptotic complexity of the algorithm is low and is quadratic in the case of quadratic objective functions.

Furthermore, we considered discrete versions of the two studied variants of resource allocation problems. While traditionally discrete resource allocations only add the restriction that the decision variable x_i should lie in $\mathbb{Z}$, the application area of DEM leads to a more restrictive variant where the feasible set for each x_i is a finite subset of $\mathbb{R}$. This makes the problem NP-hard, even if each of these sets Z_i is the same. However, in the domain of decentralized energy management, there is a natural relaxation for this problem, similar to one found in the area of DVFS (also known as speed scaling). These relaxations allow efficient solution methods while producing solutions that are very similar to solutions of the discrete problems we originally introduce. More precisely, in most practical cases we expect that the extra switching between operational values in the solutions to the relaxations is minimal, causing only minimal extra wearing of the considered devices in practice.

The models herein do not capture all (common) constraints and objectives encountered in the area of DEM. For example, another important constraint often seen is that of minimum run times. Such a constraint forces a device, for example a heat pump, to stay on for a minimum amount of time when it is switched on. This is commonly done to minimize device wearing and increase efficiency. An example of an important factor for the local objective of a device is the degradation of the device. Such constraints and objectives often add a dependency between the time intervals considered in the scheduling problem. This dependency causes the problem to become more difficult, which is left for future work.

A challenge in energy management is the unpredictability of several aspects, particularly the production from new, renewable sources and the availability of flexible devices. As the device-level problems covered herein can be solved very efficiently, we believe such uncertainty can be accounted for, by rescheduling (subsets of) devices. In case the central controller, responsible for sending steering signals, notices a (large) mismatch between a prediction and the corresponding observation (e.g. an EV arrives later or needs to depart sooner, or the solar production is more/less than expected), it can update the steering signals sent to the devices to compensate. The central controller then requests the device to reschedule based on the updated steering signal. We believe the efficient solution methods developed herein assist in ensuring that such an approach can be used in practice to also address observed mismatches between predictions and reality.

References

Barbato A, Capone A (2014) Optimization models and methods for demand-side management of residential users: a survey. Energies 7(9):5787. doi:10.3390/en7095787

Blum M, Floyd RW, Pratt V, Rivest RL, Tarjan RE (1973) Time bounds for selection. J Comput Syst Sci 7(4):448–461

Boyd S, Vandenberghe L (2004) Convex optimization. Cambridge University Press, New York

Bretthauer KM, Shetty B (2002) A pegging algorithm for the nonlinear resource allocation problem. Comput Oper Res 29(5):505–527

Claessen F, Claessens B, Hommelberg M, Molderink A, Bakker V, Toersche H, van den Broek M (2014) Comparative analysis of tertiary control systems for smart grids using the flex street model. Renew Energy 69:260–270. doi:10.1016/j.renene.2014.03.037

Claessens BJ, Vandael S, Ruelens F, Hommelberg M (2012) Self-learning demand side management for a heterogeneous cluster of devices with binary control actions. In: 3rd IEEE PES innovative smart grid technologies Europe (ISGT Europe), pp 1–8. doi:10.1109/ISGTEurope.2012.6465679

Flexiblepower alliance network (2016) EF-Pi framework. http://flexible-energy.eu/. Accessed on 27 June 2016

Gan L, Topcu U, Low SH (2013) Optimal decentralized protocol for electric vehicle charging. IEEE Trans Power Syst 28(2):940–951. doi:10.1109/TPWRS.2012.2210288

Gerards MET, Toersche HA, Hoogsteen G, van der Klauw T, Hurink JL, Smit GJM (2015) Demand side management using profile steering. In: PowerTech, IEEE Eindhoven, pp 6

Gönsch J, Hassler M (2016) Sell or store? An ADP approach to marketing renewable energy. OR Spectr 38(3):633–660

Hochbaum D, Hong SP (1995) About strongly polynomial time algorithms for quadratic optimization over submodular constraints. Math Program 69(1–3):269–309

Huang W, Wang Y (2009) An optimal speed control scheme supported by media servers for low-power multimedia applications. Multimed Syst 15(2):113–124

Hvattum LM, Norstad I, Fagerholt K, Laporte G (2013) Analysis of an exact algorithm for the vessel speed optimization problem. Networks 62(2):132–135

Kesler M, Kisacikoglu MC, Tolbert LM (2014) Vehicle-to-grid reactive power operation using plug-in electric vehicle bidirectional offboard charger. IEEE Trans Ind Electron 61(12):6778–6784

Kleinmann P, Schultz R (1990) A simple procedure for optimal load dispatch using parametric programming. Z für Oper Res 34(3):219–229. doi:10.1007/BF01415985

Kok J (2013) Planning in smart grids. Ph.D. thesis

Kwon WC, Kim T (2005) Optimal voltage allocation techniques for dynamically variable voltage processors. ACM Trans Embed Comput Syst 4(1):211–230

Li M, Yao AC, Yao FF (2006) Discrete and continuous min-energy schedules for variable voltage processors. Proc Nat Acad Sci USA 103(11):3983–3987

McKenna K, Keane A (2014) Discrete elastic residential load response under variable pricing schemes. In: IEEE PES innovative smart grid technologies, Europe, pp 1–6, doi:10.1109/ISGTEurope.2014.7028769

Mohsenian-Rad AH, Wong VWS, Jatskevich J, Schober R, Leon-Garcia A (2010) Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid. IEEE Trans Smart Grid 1(3):320–331

Molderink A, Bakker V, Bosman M, Hurink J, Smit G (2010) On the effects of mpc on a domestic energy efficiency optimization methodology. In: IEEE international energy conference and exhibition (EnergyCon), 2010, pp 120–125

Norstad I, Fagerholt K, Laporte G (2011) Tramp ship routing and scheduling with speed optimization. Transp Res Part C Emerg Technol 19(5):853–865

Patriksson M (2008) A survey on the continuous nonlinear resource allocation problem. Eur J Oper Res 185(1):1–46

Patriksson M, Strömberg C (2015) Algorithms for the continuous nonlinear resource allocation problemnew implementations and numerical studies. Eur J Oper Res 243(3):703–722

Rockafellar RT (1970) Convex analysis. Princeton University Press, Princeton

Siano P (2014) Demand response and smart grids: a survey. Renew Sustain Energy Rev 30:461–478

Tang W, Bi S, Zhang YJ (2014) Online coordinated charging decision algorithm for electric vehicles without future information. IEEE Trans Smart Grid 5(6):2810–2824

Telaretti E, Ippolito M, Dusonchet L (2016) A simple operating strategy of small-scale battery energy storages for energy arbitrage under dynamic pricing tariffs. Energies 9(1):12. doi:10.3390/en9010012

Van Den Bosch PPJ (1985) Optimal static dispatch with linear, quadratic and non-linear functions of the fuel costs. IEEE Trans Power Appar Syst PAS 104(12):3402–3408. doi:10.1109/TPAS.1985.318869

Van Den Bosch PPJ, Lootsma FA (1987) Scheduling of power generation via large-scale nonlinear optimization. J Optim Theory Appl 55(2):313–326. doi:10.1007/BF00939088

Vardakas JS, Zorba N, Verikoukis CV (2015) A survey on demand response programs in smart grids: pricing methods and optimization algorithms. IEEE Commun Surv Tutor 17(1):152–178

Vidal T, Jaillet P, Maculan N (2014) A decomposition algorithm for nested resource allocation problems. SIAM J OPTIM 26(2):1322–1340

Yao F, Demers A, Shenker S (1995) A scheduling model for reduced CPU energy. In: Proceedings of the 36th annual symposium on foundations of computer science, 1995, pp 374–382

You S, Hu J, Pedersen AB, Andersen PB, Rasmussen CN, Cha S (2012) Numerical comparison of optimal charging schemes for electric vehicles. In: 2012 IEEE power and energy society general meeting, pp 1–6. doi:10.1109/PESGM.2012.6345356